Table of Contents

Porting to Columbia

Default or Recommended compiler version and options

DRAFT

This article is being reviewed for completeness and technical accuracy.

Intel compiler versions 10.0, 10.1, 11.0 and 11.1 are available on Columbia as modules. Use the 'module avail' command to find available versions.

The current default compiler module on Columbia is *intel-comp.10.1.013*.

In addition to the few flags mentioned in the article Recommended Intel Compiler Debugging Options, here are a few more to keep in mind:

Turn on optimization: *-O3*

If you do not specify an optimization level (*-On*, *n=0,1,2,3*), the default is *-O2*. If you want more aggressive optimizations, you can use *-O3*. Note that using *-O3* may not improve performance for some programs.

Turn inlining on: *-ip* or *-ipo*

Use of *-ip* enables additional interprocedural optimizations for single file compilation. One of these optimizations enables the compiler to perform inline function expansion for calls to functions defined within the current source file.

Use of *-ipo* enables multifile interprocedural (IP) optimizations (between files). When you specify this option, the compiler performs inline function expansion for calls to functions defined in separate files.

Parallelize your code: *-openmp* or *-parallel*

-openmp handles OMP directives and *-parallel* looks for loops to parallelize.

For more compiler/linker options, read **man ifort, man icc**, or

```
%ifort -help
%icc -help
```

Porting to Columbia: With SGI's MPT

DRAFT

This article is being reviewed for completeness and technical accuracy.

The available SGI MPT modules on Columbia are:

```
mpt.1.16.0.0
mpt.1.18.0.0
mpt.1.19.0.0
mpt.1.22.0.0
mpt.1.25
```

The current default version is *mpt.1.16.0.0*.

Environment Variables

On Columbia, when you load any of the above MPT modules, several environment variables such as CPATH, INCLUDE, LD_LIBRARY_PATH, etc., are modified by pre-pending the appropriate MPT directories. Also, the following MPT-related environment variables are modified from their default values for improved performance:

```
setenv MPI_BUFS_PER_HOST 256
setenv MPI_BUFS_PER_PROC 256
setenv MPI_DSM_DISTRIBUTE
```

The meanings of these variables and their default values are:

- MPI_BUFS_PER_HOST

  Determines the number of shared message buffers (16 KB each) that MPI is to allocate for each host (i.e., C21, C22, C23, C24). These buffers are used to send and receive long inter-host messages.

  Default: 32 pages (1 page = 16KB) for mpt.1.16, mpt.1.18, mpt.1.19, mpt.1.22
  Default: 96 pages (a page = 16KB) for mpt.1.25

- MPI_BUFS_PER_PROC

  Determines the number of private message buffers (16 KB each) that MPI is to allocate for each process. These buffers are used to send long messages and intra-host messages.

  Default: 32 pages (1 page = 16KB)

- MPI_DSM_DISTRIBUTE (toggle)

  Activates NUMA job placement mode. This mode ensures that each MPI process gets a unique CPU and physical memory on the host with which that CPU is associated. This feature can also be overridden by using dplace or omplace. This feature is most useful if running on a dedicated system or running within a cpuset.

  Default: Not enabled

Building Applications

Building MPI applications with SGI's MPT library simply requires linking with -lmpi and/or -lmpi++. See the article SGI MPT for some examples.

Running Applications

```
MPI executables built with SGI's MPT are not allowed to run on the
Columbia front-end node.
```

You can run your MPI job on C21 - C24 in an interactive PBS session or through a PBS batch job. Use **mpiexec** (under /PBS/bin) or mpirun to start your MPI processes. For example:

```
#PBS -lncpus=8
....
mpiexec -np N ./your_executable
```

The -np flag (with *N* MPI processes) can be omitted if the value of *N* is the same as the value specified for *ncpus*.

Porting to Columbia: With OpenMP

DRAFT

This article is being reviewed for completeness and technical accuracy.

Building Applications

To build an OpenMP application, you need to use the -openmp Intel compiler flag:

```
%ifort -o your_executable -openmp program.f
```

Note that if you are compiling separate files, then -openmp is required at the link step to link in the OpenMP library.

Running Applications

Note that *OMP_NUM_THREADS* is set to 1 by default for PBS jobs. Reset it to the number of threads that you want.

Here is a sample PBS script for running OpenMP applications on Columbia:

```
#PBS -lncpus=8,walltime=1:00:00

setenv OMP_NUM_THREADS 8

cd $PBS_O_WORKDIR

./your_executable
```

Porting to Columbia: With MPI and OpenMP

DRAFT

This article is being reviewed for completeness and technical accuracy.

Building Applications

To build a hybrid MPI+OpenMP application, you need to compile your code with the -openmp compiler flag and link in both the Intel OpenMP and the SGI MPT library:

```
%ifort -o your_executable -openmp program.f -lmpi
```

Running Applications

Process/thread placement is critical to the performance of MPI+OpenMP hybrid codes. Two environment variables should be set to get the proper placement:

- MPI_DSM_DISTRIBUTE

  Activates NUMA job placement mode. This mode ensures that each MPI process gets a unique CPU and physical memory on the node with which that CPU is associated. Currently, the CPUs are chosen by simply starting at relative CPU 0 and incrementing until all MPI processes have been forked.

- MPI_OPENMP_INTEROP

  Setting this variable modifies the placement of MPI processes to better accommodate the OpenMP threads associated with each process. For this variable to take effect, you must also set MPI_DSM_DISTRIBUTE.

Also note that *OMP_NUM_THREADS* is set to 1 by default for PBS jobs. Reset it to the number of threads that you want.

Here is a sample PBS script for running MPI+OpenMP hybrid (2 MPI processes, 4 OpenMP threads per MPI process) applications on Columbia:

```
#PBS -lncpus=8,walltime=1:00:00

setenv MPI_DSM_DISTRIBUTE
setenv MPI_OPENMP_INTEROP
setenv OMP_NUM_THREADS 4

cd $PBS_O_WORKDIR

mpirun -np 2 ./your_executable
```